CRoC: Context Refactoring Contrast for Graph Anomaly Detection with Limited Supervision

Siyue Xie[1,*], **Da Sun Handason Tam**[1] **and Wing Cheong Lau**[1]

[1]The Chinese University of Hong Kong
xiesiyue@link.cuhk.edu.hk, handasontam@gmail.com, wclau@ie.cuhk.edu.hk

Abstract. Graph Neural Networks (GNNs) are widely used as the engine for various graph-related tasks, with their effectiveness in analyzing graph-structured data. However, training robust GNNs often demands abundant labeled data, which is a critical bottleneck in real-world applications. This limitation severely impedes progress in Graph Anomaly Detection (GAD), where anomalies are inherently rare, costly to label, and may actively camouflage to evade detection. To address these problems, we propose **C**ontext **R**efact**o**ring **C**ontrast (CRoC), a simple yet effective framework that trains GNNs for GAD by jointly leveraging limited labeled and abundant unlabeled data. Unlike previous works, CRoC exploits the class imbalance inherent in GAD to refactor the context of each node, which builds augmented graphs by recomposing the attributes of nodes while preserving their interaction patterns. Furthermore, CRoC encodes heterogeneous relations separately and integrates them into the message-passing process, inducing the model to capture complex interaction semantics. These operations preserve node semantics while encouraging robustness against adverse camouflage, enabling GNNs to uncover intricate anomalous cases. In the training stage, CRoC is further integrated with the contrastive learning paradigm. This allows GNNs to effectively harness unlabeled data during joint training, producing richer, more discriminative node embeddings. CRoC is evaluated on seven real-world GAD datasets with different sizes. Extensive experiments demonstrate that CRoC achieves up to 14% AUC improvement over baseline GNNs and outperforms state-of-the-art GAD methods under limited-label settings.

1 Introduction

Graph Anomaly Detection (GAD) is a task aiming to identify nodes that deviate significantly from the majority in terms of roles or behaviors within a graph [18]. These instances, though rare, can have a profound impact on the entire system. Therefore, GAD has drawn considerable attention across domains, with a wide range of applications, such as fraudster detection [32], malicious accounts mining [37], spam review filtering [6, 17], etc.

To detect graph anomalies, Graph Neural Networks (GNNs) [14, 10] have emerged as a leading solution for GAD. Generally, GNNs learn node embeddings by aggregating neighborhood information of the target node. By iteratively stacking multiple layers, GNNs can encode useful context information critical for a downstream task.

However, there are still many obstacles when applying GNNs for GAD. A primary concern is that GNNs are typically trained in a supervised manner, relying heavily on labeled data. Training GNNs with very few labeled data may lead to severe overfitting and degraded performance. However, in real-world GAD scenarios, informative labeled data are often scarce due to the fact that: (1) the target of interest, i.e., the anomalous nodes, are usually the minority in the graph, and (2) sometimes only experienced human experts are qualified to annotate, making it costly to collect adequate training labels. Moreover, the class distribution in GAD is extremely imbalanced since normal nodes account for the majority. Ignoring this nature may result in biased predictions. Another concern is that typical GNNs work better in homophily graphs, where adjacent nodes are expected to be in the same class. However, in GAD, it is common to find anomalous nodes connected to normal nodes, resulting in heterophilic connections. This is because anomalous nodes, e.g., fraudsters in payment networks, may mimic normal nodes' features or interact with normal nodes to camouflage themselves. Some previous works on GAD resort to pruning heterophilic edges [9, 20] or sampling homophilic edges [6, 16] for GNN models. However, such operations can be defective since GNNs are not inherently designed to react proactively against camouflage in the graph.

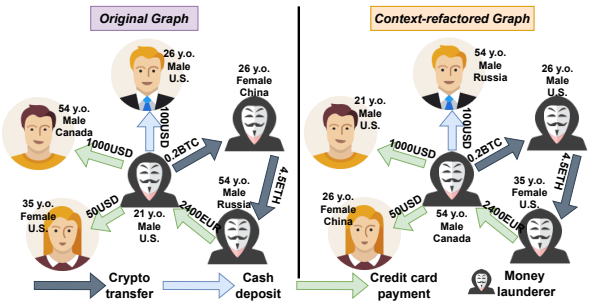

Figure 1. A toy example of context refactoring: randomly shuffle user profiles in a transaction network. The role of the centered anomalous node ought to be consistent if edge information is preserved.

To address these challenges, we propose Context Refactoring Contrast (CRoC) to enhance GNNs for GAD when training labels are limited. Considering the prevalence of camouflage in real-world GAD scenarios, we refactor the context of nodes by deliberately introducing (simulated) camouflage into the graph to train a more robust model. ('Context' refers to any information about a node's neighborhood, including node/ edge attributes, graph topologies, etc.) An illustrative example is shown in Fig. 1. Concretely, we pro-

* Corresponding Author.

pose to refactor a graph by shuffling the node features but preserving the relations/ interactions of edges, which is expected to keep the semantics of a node consistent. This is particularly effective for GAD scenarios since it utilizes the prior knowledge of class imbalance and (the existence of) camouflage in the graph. Anomalous nodes with feature camouflage are observed to have node features similar to those of normal nodes [6]. Since normal nodes account for the majority of the graph, shuffling node features hardly changes the semantics of normal and camouflaged anomalous nodes, as their context changes little. On the other hand, anomalous patterns are highly correlated with edge information, which encodes special relations and interaction behaviors between nodes. Preserving the edge information of the original graph for context refactoring essentially maintains most of the critical information required for GAD. Through context refactoring, features of unlabeled nodes can also be reused in the context of the labeled nodes, which effectively correlates the labeled and unlabeled data. In addition, we propose a relation-aware joint aggregation operator, enabling GNNs to distinguish diverse interaction patterns. From the perspective of self-supervised learning, the context-refactored graph is an augmentation of the original graph in the view of semantics. Therefore, it is natural to integrate context refactoring with contrastive learning. A node-wise contrast is then applied between the original and the context-refactored graph. With contrastive learning, all unlabeled nodes are actively incorporated into the training process, encouraging the model to learn from the abundant unlabeled data and yielding more comprehensive and robust node representations for GAD tasks. We evaluate CRoC across seven real-world GAD datasets with only limited labels provided for training. Extensive experiments show that CRoC can effectively enhance GNN models (up to 14% improvement w.r.t AUC) and outperforms other state-of-the-art GAD schemes. Our code[1] and appendix[2] are publicly available.

We summarize our contributions as follows:

- We analyze some common challenges of GAD and propose context refactoring, which effectively correlates labeled and unlabeled data to train GNN against camouflage with limited supervision.
- We integrate context refactoring into the contrastive learning paradigm and propose a relation-aware joint aggregation operator, enabling a GNN to learn more comprehensive and discriminative representations for GAD tasks.
- We evaluate our scheme on seven real-world GAD datasets. Experimental results demonstrate CRoC's superiority when only limited labels are available for training.

2 Related Works

2.1 Graph Neural Networks

Graph Neural Networks (GNNs) are a family of models for dealing with graph-structured data. Pioneer researchers design GNN kernels based on graph spectrum theory [1], while follow-up works simplify the operator by a message-passing framework [14, 10] and achieve better performances [36, 27]. Similar to other deep-learning models, GNNs are typically trained in a supervised manner. There are also works attempting to enhance GNNs by learning from unlabeled data. One typical way is through self-training [2, 23], which picks confident predictions to extend the training set. Another set of works pre-trains a GNN through contrastive learning [26, 28], where handcrafted augmentations are designed to form contrastive pairs for unsupervised training. Others [7, 33, 34] resort to training GNNs together with well-designed self-supervised objectives, aiming to align learned embeddings to a downstream task.

2.2 Graph Anomaly Detection with GNNs

With great effectiveness in processing graph data, GNNs are elaborately adapted to handle GAD tasks as per different challenges. Some works utilize GAD's class imbalance (i.e., normal nodes are the majority of the graph) for detecting anomalies. They train GNNs to reconstruct a graph, where normal nodes are assumed to be better reconstructed. Instances with relatively poor reconstruction results are flagged as anomalous [21, 11]. Some methods rely on the proximity hypothesis to train their detector, where neighboring nodes' representations are assumed to be similar to each other. A node is supposed to be anomalous if its representation is dissimilar to its neighbors' [20, 13]. Researchers also notice that camouflage will result in non-homophilic connections, which are undesired for typical GNNs. To deal with it, several approaches [6, 15, 16] propose selective node sampling during aggregation to reduce heterophilic interference. BWGNN [24] and SplitGNN [30] handle GAD by applying well-designed graph kernels with tunable frequencies. GHRN [9] prunes heterophilic edges to build a new graph for training, while GDN [8] and GTAN [31] mine heterophily-robust features for the model. Another group of methods [29, 4, 38] attempts to enhance GNNs by introducing graph augmentations and self-supervised learning techniques. However, many aforementioned works assume that they have abundant labeled anomalous data to train their GNN backbones or specialized modules, such as the node-wise similarity predictor in CARE-GNN or the edge heterophily indicator in GHRN. The effectiveness of these modules can be challenged when labels for training are very limited, which is common in real-world GAD scenarios. More discussions on related works can be found in Appendix A [35].

3 Preliminary Analysis and Motivations

3.1 Problem Formulations

We formulate GAD as a semi-supervised node classification task: given a multi-relation graph $\mathcal{G} = (\mathcal{V}, \boldsymbol{X}, \{\mathcal{E}_r\}|_{r=1}^{R})$, where $\mathcal{V} = \{v_i\}|_{i=1}^{N}$ is a N-sized node set and $\boldsymbol{X} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_N]^\intercal \in \mathbb{R}^{N \times d_0}$ is the corresponding node feature matrix. The edge set $\{\mathcal{E}_r\}|_{r=1}^{R}$ encompasses R relations. Each edge $e_{i,j}^r = (v_i, v_j) \in \mathcal{E}_r$ is associated with a relation r. $\mathcal{V}$ consists of two subsets $\mathcal{V} = \{\mathcal{V}^L, \mathcal{V}^U\}$, which denotes the labeled ($\mathcal{V}^L$) and unlabeled ($\mathcal{V}^U$) node set respectively. Let $\mathbf{Y}^L = \{\boldsymbol{y}_v | v \in \mathcal{V}^L\}$ be the labels for $\mathcal{V}^L$, where $\boldsymbol{y}_v$ is a one-hot vector to indicate the class of node v. Note that in GAD with limited supervision, $|\mathcal{V}^L| \ll N$. Our goal is to train a GNN-based model given $\mathcal{G}$ and $\mathbf{Y}^L$ to predict the class (normal/ anomalous) of all unlabeled nodes.

3.2 Challenges of Camouflage in Graphs

Camouflage poses a significant challenge in GAD tasks, which has drawn great attention in some previous works [6, 39]. Typically, there are two kinds of camouflage in GAD. One is node feature camouflage, where anomalous nodes deliberately mimic their node features like normal nodes. For example, fraudsters may declare false personal information in social networks, rendering their profile (node features) less reliable for GAD. Another type is behavior camouflage,

[1] https://github.com/XsLangley/CRoC_ECAI2025
[2] https://arxiv.org/abs/2508.12278

Table 1. Statistics of two typical GAD datasets.

Dataset	#Node	#Edge	#Anomaly	Anomaly(%)
T-Soc	5.7M	73M	174,280	3.01%
DGraph	3.7M	4.3M	15,509	0.42%

Table 2. Averaged node coverage of a typical GNN on T-Soc with variant depth and label rate settings.

label rate	0.01%	0.05%	0.1%
1-layer	0.25%	1.25%	2.48%
2-layer	8.73%	22.12%	31.47%
3-layer	33.82%	56.53%	67.77%

where anomalous nodes engage in both malicious and benign interactions simultaneously. Such malicious interactions can be informative for GAD. However, these critical clues may be overwhelmed by accompanying normal interactions if all interactions are equally treated. Therefore, to avoid being misled by camouflage, we should train a GNN to be robust against feature camouflage and endow it with the ability to distinguish different kinds of interactions in the graph.

3.3 Limited Supervision for GNNs in GAD Tasks

GNNs' performance is highly correlated with the amount of labeled data provided for training. However, in real-world scenarios, supervised information (labels) is very limited due to the great difficulty and high expense of annotation. In GAD, this challenge is exacerbated by the naturally imbalanced class distribution, where our target of interest, i.e., anomalous nodes, are always the minority. We show the statistics of two real-world GAD datasets in Table 1 as an example. Suppose 1% of nodes could be annotated for training, the desired supervised information (labeled anomalies) remains scarce even for large-scale datasets. For example, in DGraph, only around 155 anomalous nodes can be labeled for training in this case. In other words, GNNs can only learn very limited knowledge about anomalies from labeled data, leading to inferior representations for GAD.

On the other hand, many unlabeled nodes remain underutilized when training GNNs following standard semi-supervised learning paradigms [14, 27]. To quantify this, **we define the node coverage of training as: $\frac{N_L^k}{N}$, where N_L^k is the number of nodes within k-hop/layer neighbourhood of all labeled training nodes.** Table 2 shows the statistics of typical message-passing GNNs (e.g., GCN and GAT). We can observe that less than 70% of nodes participate in training, even if we stack a 3-layer GNN model with a 0.1% label rate setting (which is equivalent to providing up to 5781 labeled nodes for training). Coverage drops significantly if given lower annotation budgets. In other words, many unlabeled nodes are essentially underexplored. However, there can be useful patterns that occur only within unlabeled nodes. The training data can hardly match the distribution of all data with such limited supervision, which may make it fail to generalize to the testing environment. Therefore, besides training GNNs with the labeled data, we should manage to utilize the abundant unlabeled data to learn more comprehensive node embeddings for better generalization ability.

4 Proposed Method

To deal with the challenges discussed in the previous section, we propose **C**ontext **R**efact**o**ring **C**ontrast (CRoC) for GAD tasks under limited supervision. The framework of CRoC is shown in Fig. 2.

4.1 Context Refactoring for GAD

As discussed in the previous section, node feature camouflage may mislead the model when making decisions based on node features. Previous works have attempted to detect such camouflage by measuring the feature similarity or disparity between node [6, 15]. However, this is unreliable as features of anomalous nodes can be the same as normal nodes, e.g., a money launderer may deliberately declare his/her user information (node features) the same as others. This easily collapses feature-based detectors, where we show an illustrative example in the visualization results (in Fig. 3). In this work, instead of endeavoring to detect camouflages, we go in the opposite direction: we proactively introduce node feature camouflage to refactor the graph. We train a GNN on the refactored graph, forcing it to learn to be robust against node feature camouflage.

To this end, an intuitive way is to replace the features of a labeled anomalous node with those of normal nodes. In CRoC, we adopt a more comprehensive strategy by refactoring the context of each node. Formally, we define the context refactoring as follows:

$$\tilde{\boldsymbol{X}} = \Omega(\boldsymbol{X}) \tag{1}$$

where $\Omega(\cdot)$ is a random shuffle function that permutes the original feature matrix $\boldsymbol{X}$ by row (i.e., we exchange the row vectors in the matrix), and $\tilde{\boldsymbol{X}} \in \mathbb{R}^{N \times d_0}$ is the feature matrix of the context-refactored graph. We argue that Equ (1) generally keeps the semantics of most nodes, which is especially effective for GAD by explicitly leveraging the class imbalance premise . On the one hand, normal nodes always account for the majority in a graph. Therefore, a normal node will be assigned node features of the other normal node with a high probability, which has minimal impact on its overall contextual semantics. On the other hand, undesired shuffle results (i.e., a normal node is assigned an anomalous node's features) are tolerable for a GNN. This is because GNNs rely not only on the information of the target node but also on its neighborhood context for decision-making. In our experiments, features will also be reshuffled in each training epoch, preventing the case where a normal node is always assigned undesired features. For anomalous nodes, they are assigned normal nodes' features with a high probability, simulating scenarios where node feature camouflage occurs. Training a GNN on a graph with explicit camouflage will force the model to learn to be more robust against camouflage. Therefore, instead of designing specific modules to alleviate the impact of node feature camouflage and class imbalance, we proactively harness these priors to boost the model.

Furthermore, considering cases where native node features are critical for detection, we propose mixing the original and shuffled node feature matrices to enhance the generalization ability:

$$\boldsymbol{X'} = \alpha \boldsymbol{X} + (1-\alpha)\tilde{\boldsymbol{X}} \tag{2}$$

where $\alpha \in [0, 1)$ is a hyper-parameter to tune the importance of native node features for context refactoring. The finalized context-refactored graph is: $\mathcal{G}' = (\mathcal{V}, \boldsymbol{X'}, \{\mathcal{E}_r\}|_{r=1}^{R})$. This Mixup [40] style augmentation enables CRoC to preserve node-specific identity information while introducing controllable variations to the graph. More importantly, unlike the original Mixup scheme, our method, as shown in Equ (2), actively incorporates unlabeled nodes into training, enabling the GNN backbone to learn more diverse patterns in GAD. More discussions refer to Appendix A.2 and B.1 [35].

4.2 Relation-aware Joint Aggregation (RJA)

In context refactoring, we shuffle node features but preserve the interaction connections (i.e., relations and edges), as interactions reflect

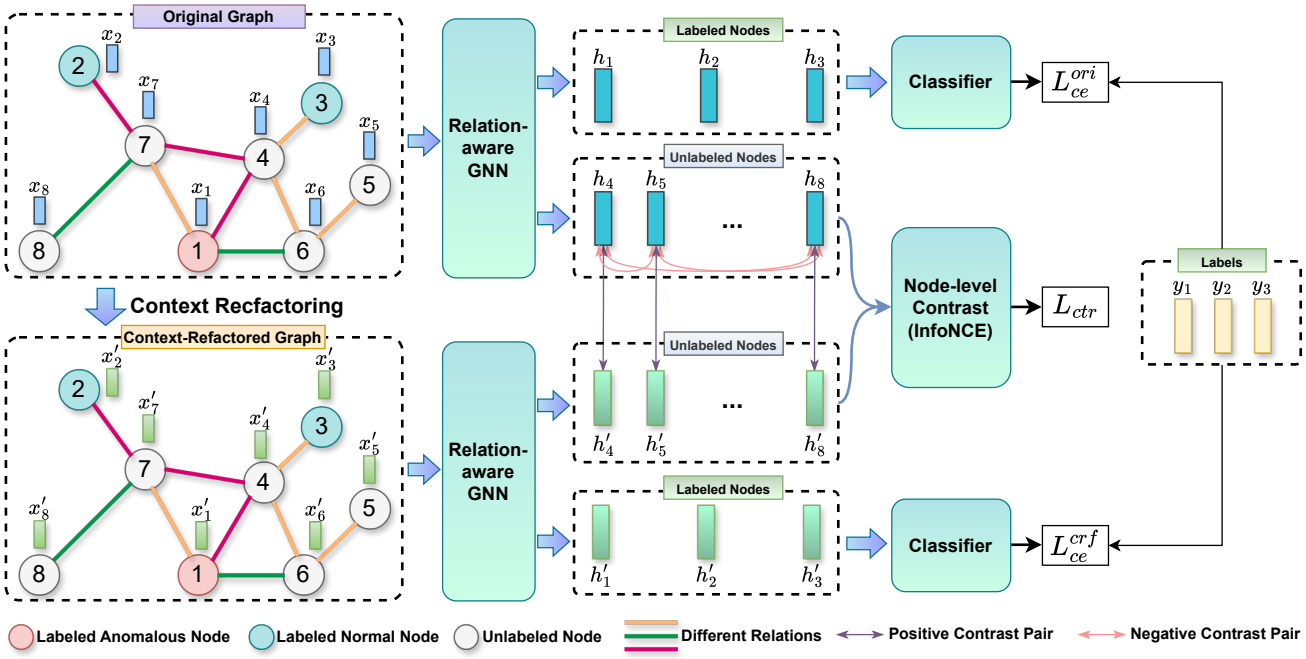

Figure 2. The framework of **C**ontext **R**efact**o**ring **C**ontrast (CRoC). Given a multi-relation graph, we refactor the original graph utilizing the class imbalance prior of GAD. Embeddings of the few labeled nodes from two different views (graphs) are supervised by ground-truth annotations, while unlabeled nodes are included for joint training via a self-supervised objective.

critical anomalous patterns. However, as discussed in Section 3.2, anomalous instances may conceal malicious interactions within a multitude of benign interactions for behavior camouflage. To handle it, some works propose to aggregate information from each relation individually within each layer [6, 15] or decouple the multi-relation graph into multiple single-relation graphs [24]. However, when a malicious activity involves multiple types of interactions/ relations, these schemes may disrupt the chain of interactions or lose track of some critical anomalous patterns. An example is shown in Fig. 1, where the money laundry chain involves two payment schemes.

To address this problem, we propose a Relation-aware Joint Aggregation (RJA) operator to learn across different relations/ interactions in multi-relation graphs. Specifically, we learn a relation embedding for each relation type in each GNN layer. Denote $\boldsymbol{h}^l_{r_{(u,v)}}$ as the learnable relation embedding corresponding to edge (u, v) in the l-th GNN layer, where $r_{(u,v)} \in \{1, 2, ..., R\}$ indexes the relation type of edge (u, v). We reformulate the aggregation operator of a message-passing GNN as follows:

$$\tilde{\boldsymbol{h}}^l_u = \sigma\left(\boldsymbol{W}^l(\boldsymbol{h}^{l-1}_u \| \boldsymbol{h}^l_{r_{(u,v)}}) + \boldsymbol{b}^l\right) \tag{3}$$

$$\boldsymbol{h}^l_v = \Phi\left(\boldsymbol{h}^{l-1}_v, Aggr(\{\tilde{\boldsymbol{h}}^l_u\}); u \in \mathcal{N}_v\right) \tag{4}$$

where $\boldsymbol{h}^l_v$ is the embedding of node v in the l-th layer, $\|$ is the concatenation operator, $\boldsymbol{W}^l$ and $\boldsymbol{b}^l$ are learnable weight and bias in the l-th layer. $\sigma(\cdot)$, $Aggr(\cdot)$ and $\Phi(\cdot)$ are activation, aggregation and transformation functions, which vary across different GNN backbones. $\mathcal{N}_v$ is the neighbor set of the target node v, with neighbor u connected via relation type $r_{(u,v)}$. Equ (3) explicitly associates a relation with the neighbor interacting with the target node, enabling a GNN to investigate cross-relational information via Equ (4). RJA explicitly distinguishes different types of relations and encourages a GNN to learn mutual correlations among them. Experimental results (Section 5.3 and 5.4) also demonstrate its effectiveness.

4.3 *Learn from the Context-Refactored Graph*

Given that the semantics of the context-refactored graph should align with the original graph, we can train a GNN in a supervised manner on both graphs using the available labels. Denote $g(\mathcal{G}^*; \theta_G)$ as a GNN parameterized by θ_G with an input graph $\mathcal{G}^*$ and $f(\boldsymbol{H}^*; \theta_C)$ as the classifier parameterized by θ_C with input $\boldsymbol{H}^*$. In our experiments, $g(\mathcal{G}^*; \theta_G)$ is implemented by GIN [36] or GraphSAGE [10], while a 2-layer MLP is used for $f(\boldsymbol{H}^*; \theta_C)$. Node embeddings from two views can be generated by feeding these two graphs (the original and the context-refactored graph) into the same GNN:

$$\boldsymbol{H} = g(\mathcal{G}\,; \theta_G), \quad \boldsymbol{H}' = g(\mathcal{G}'; \theta_G) \tag{5}$$

where $\boldsymbol{H} = [\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_N]^\intercal$ and $\boldsymbol{H}' = [\boldsymbol{h}'_1, \boldsymbol{h}'_2, ..., \boldsymbol{h}'_N]^\intercal$ are $\mathbb{R}^{N\times d}$ embedding matrices w.r.t the original and the refactored graph, with d the embedding dimension. The classifier's predictions are formulated as:

$$\hat{\boldsymbol{Y}} = f(\boldsymbol{H}; \theta_C), \quad \hat{\boldsymbol{Y}}' = f(\boldsymbol{H}'; \theta_C) \tag{6}$$

where $\hat{\boldsymbol{Y}} = [\hat{\boldsymbol{y}}_1, \hat{\boldsymbol{y}}_2, ..., \hat{\boldsymbol{y}}_N]^\intercal$ and $\hat{\boldsymbol{Y}}' = [\hat{\boldsymbol{y}}'_1, \hat{\boldsymbol{y}}'_2, ..., \hat{\boldsymbol{y}}'_N]^\intercal$ represent the normalized prediction confidence matrices. We can formulate the objectives over the set of labeled nodes by the cross-entropy loss:

$$L^{ori}_{ce} = -\frac{1}{|\mathcal{V}^L|}\sum_{v\in\mathcal{V}^L} \boldsymbol{y}^\intercal_v \log(\hat{\boldsymbol{y}}_v) \tag{7}$$

$$L^{crf}_{ce} = -\frac{1}{|\mathcal{V}^L|}\sum_{v\in\mathcal{V}^L} \boldsymbol{y}^\intercal_v \log(\hat{\boldsymbol{y}}'_v) \tag{8}$$

By jointly optimizing Equ (7) and (8), we can train a GNN in a supervised manner.

It is worth noting that L_{ce}^{crf} enables GNNs to access the abundant unlabeled data and correlate it with the limited labeled data during training. This is because features of unlabeled nodes can be shuffled to the neighborhood of a labeled node, thereby contributing to L_{ce}^{crf} by mixing with the original features by Equ (2). This operation enables GNNs to efficiently reuse the context information of labeled nodes (by mixing up different node features). This improves the input diversity and implicitly reduces the risk of overfitting limited labeled data. Moreover, unlike methods that learn from unlabeled data via pseudo-labels [4, 23], L_{ce}^{crf} is fully supervised by ground-truth labels. This avoids the pitfalls of learning from incorrect knowledge (raised by noisy pseudo-labels) and thereby generates more discriminative embeddings. More analysis refers to Appendix B.1 [35].

4.4 Context Refactoring Contrast

From the perspective of data augmentation, context refactoring is a graph augmentation scheme specific to GAD tasks. This makes it a natural fit for integration with contrastive learning paradigms.

Specifically, we apply a node-wise contrast scheme for training. As shown in Fig. 2, we collect unlabeled nodes' embeddings from the two different views (the original and the context-refactored graphs). Since context refactoring preserves the context semantics, we pair an unlabeled node's embedding from the original graph with its counterpart from the context-refactored graph to form a positive sample. For negative samples, we use an intra-view approach by randomly selecting embeddings of different nodes from the original graph. We finally contrast the positive against negative samples following the InfoNCE loss [12]:

$$L_{ctr} = -\frac{1}{|\mathcal{V}^U|} \sum_{v, u_i \in \mathcal{V}^U} \log \frac{exp(\boldsymbol{h}_v^\intercal \boldsymbol{h}_v'/\tau)}{\sum_{u_i \neq v}^{i=1\sim k} exp(\boldsymbol{h}_v^\intercal \boldsymbol{h}_{u_i}/\tau)} \tag{9}$$

where τ is a tunable temperature and k is the number of negative samples for each positive sample.

Note that Equ (9) incorporates all unlabeled nodes in the objective function. This enables a GNN to learn from the neighborhood context of the massive unlabeled nodes, inducing the model to capture more diverse patterns for generating more powerful node embeddings. Additional analyses refer to Appendix B.2 [35].

4.5 Train GNNs with CRoC

We enhance a GNN-based GAD system via CRoC by optimizing the following loss function:

$$L_{all} = L_{ce}^{ori} + \gamma L_{ce}^{crf} + \eta L_{ctr} \tag{10}$$

where γ and η are two hyper-parameters to balance the objective. Note that in Equ (10), we unify the self-supervised objective (L_{ctr}) and the supervised parts (L_{ce}^{ori} and L_{ce}^{crf}) for joint training in an end-to-end manner. Therefore, more diverse patterns can be learned via L_{ctr} to supplement the limited supervised information for decision making. Simultaneously, L_{ctr} is regulated by the supervised signal provided by L_{ce}^{ori} and L_{ce}^{crf}, ensuring the model to learn extra knowledge that will not deviate from the target of the task. Therefore, CRoC bridges the few labeled data and abundant unlabeled data, enabling GNNs to learn more powerful representations for GAD tasks under the guidance of scarce supervised signals.

5 Experimental Evaluations

5.1 Experimental Setup

5.1.1 Datasets

We evaluate CRoC across seven real-world GAD datasets, encompassing diverse domains such as commodity review fraud (Amazon and Yelp), financial fraud accounts (T-Fin and DGraph) and anomalies in social networks (T-Soc, Weibo and Reddit). Detailed descriptions and statistics for all datasets are provided in Appendix C.1 [35].

5.1.2 Baselines

We compare CRoC with three groups of methods for GAD tasks: **(1) Common feature-based classifiers:** MLP and XGBoost (XGB) [5]. **(2) GNN baselines:** GCN [14], GAT [27], GIN [36], SAGE [10] and RGCN [22]. **(3) Specialized GAD models:** CARE-GNN [6], DCI [29], PCGNN [16], BWGNN [24], XGBGraph [25], GHRN [9] and ConsisGAD [4]. Methods [39, 8, 15] that do not report better results than the listed models are omitted for comparison.

5.1.3 Implementations

We follow BWGNN's settings [24] to split datasets: 1% (0.01% for T-Soc) of nodes are randomly sampled to form the training set (labeled nodes), while the remaining are split into validation and test sets in a proportion of $1:2$. We implement CRoC by a 2-layer GIN (or SAGE) with 64 hidden dimensions. We set $\tau = 2$, search α, γ and η in $[0, 1]$ and train GNNs with learning rate 0.003. In experiments, we build a new context-refactored graph for every training epoch. The model is finally evaluated on the original graph. We run each model 10 rounds with different random seeds and record the test result w.r.t. the best validation results for each round. We use the Area under the ROC Curve (AUC) and the Average Precision score (AP) to measure performance and the reported value is averaged over ten runs. More details refer to Appendix C.2 [35].

5.2 Results on GAD with Limited Supervision

We show the results in Table 3. CRoC outperforms all other methods in the three largest datasets (i.e., T-Soc, DGraph and Yelp) and is comparable to the state-of-the-art in Amazon and T-Fin. We draw the following conclusions from the experiment observation:

- Context refactoring is effective in enhancing the GNN backbone. By training on the context-refactored graph, a GNN is forced to be robust against potential node feature camouflage in the graph. The relation-aware joint aggregation distinguishes different kinds of relations, which helps detect behaviour camouflage. These mechanisms are particularly effective in Yelp and T-Soc: the former is known to have reviews with camouflage, while the latter, sourced from social networks, includes node feature camouflage. This enables our model to outperform camouflage-unaware models by a large margin.
- Models trained relying on supervised information may fail to learn sufficient knowledge from such limited labeled data. For example, CARE-GNN needs labeled nodes to train its similarity measurement, and GHRN requires labels for edge-pruning. In contrast, CRoC leverages both limited labeled data and abundant unlabeled data, empowering the model to capture more diverse patterns than other methods.

Table 3. Experimental Results on five GAD Datasets. The label rate of each dataset is marked in parentheses.

Dataset	T-Soc (0.01%)		DGraph (1%)		Yelp (1%)		T-Fin (1%)		Amazon (1%)	
Metric	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP
MLP	$69.49_{\pm 3.06}$	$6.96_{\pm 1.29}$	$71.20_{\pm 0.24}$	$2.55_{\pm 0.06}$	$72.43_{\pm 0.85}$	$31.61_{\pm 1.71}$	$90.02_{\pm 1.12}$	$58.47_{\pm 2.94}$	$89.20_{\pm 1.39}$	$53.85_{\pm 5.78}$
XGB	$66.67_{\pm 2.04}$	$6.35_{\pm 0.98}$	$68.79_{\pm 0.40}$	$2.39_{\pm 0.05}$	$75.96_{\pm 1.13}$	$38.20_{\pm 2.29}$	$92.39_{\pm 0.50}$	$73.96_{\pm 3.48}$	$91.29_{\pm 4.31}$	$72.83_{\pm 10.75}$
GCN	$81.85_{\pm 2.07}$	$18.58_{\pm 6.32}$	$69.25_{\pm 1.09}$	$2.55_{\pm 1.09}$	$55.81_{\pm 1.15}$	$18.23_{\pm 1.21}$	$91.55_{\pm 0.66}$	$73.73_{\pm 1.41}$	$79.25_{\pm 1.79}$	$26.33_{\pm 4.51}$
RGCN	$81.85_{\pm 3.30}$	$15.75_{\pm 1.67}$	$69.97_{\pm 0.87}$	$2.77_{\pm 0.11}$	$59.27_{\pm 3.93}$	$18.25_{\pm 1.58}$	$87.59_{\pm 1.29}$	$44.64_{\pm 3.67}$	$61.36_{\pm 3.65}$	$10.68_{\pm 1.23}$
GAT	$89.52_{\pm 1.18}$	$33.12_{\pm 8.89}$	$68.68_{\pm 1.38}$	$2.64_{\pm 0.16}$	$55.68_{\pm 0.99}$	$18.31_{\pm 0.92}$	$91.87_{\pm 0.74}$	$77.30_{\pm 0.92}$	$83.47_{\pm 2.12}$	$36.86_{\pm 7.01}$
SAGE	$81.58_{\pm 1.95}$	$17.65_{\pm 7.12}$	$73.09_{\pm 0.57}$	$3.06_{\pm 0.12}$	$72.88_{\pm 0.82}$	$32.64_{\pm 1.22}$	$92.36_{\pm 1.01}$	$76.90_{\pm 1.42}$	$88.03_{\pm 2.51}$	$56.70_{\pm 5.40}$
GIN	$87.86_{\pm 4.97}$	$21.96_{\pm 5.92}$	$71.13_{\pm 0.89}$	$2.75_{\pm 0.16}$	$73.28_{\pm 1.26}$	$32.78_{\pm 2.26}$	$90.79_{\pm 1.04}$	$56.90_{\pm 7.04}$	$84.61_{\pm 2.05}$	$43.83_{\pm 8.63}$
XGBGraph	$87.20_{\pm 2.38}$	$54.90_{\pm 2.31}$	$67.02_{\pm 0.76}$	$2.30_{\pm 0.07}$	$74.76_{\pm 0.95}$	$36.49_{\pm 1.69}$	$93.75_{\pm 3.24}$	$81.42_{\pm 3.56}$	$91.42_{\pm 3.05}$	$72.29_{\pm 10.53}$
CARE-GNN	$69.81_{\pm 4.13}$	$6.30_{\pm 1.19}$	$69.25_{\pm 0.59}$	$2.18_{\pm 0.11}$	$73.70_{\pm 2.68}$	$33.30_{\pm 3.25}$	$89.51_{\pm 0.95}$	$56.81_{\pm 5.18}$	$88.52_{\pm 1.73}$	$49.54_{\pm 6.43}$
DCI	$82.12_{\pm 0.86}$	$13.48_{\pm 1.59}$	$70.00_{\pm 0.52}$	$2.65_{\pm 0.07}$	$65.71_{\pm 7.22}$	$25.41_{\pm 6.34}$	$91.15_{\pm 0.63}$	$73.79_{\pm 2.10}$	$88.75_{\pm 1.62}$	$49.66_{\pm 7.95}$
BWGNN	$86.27_{\pm 2.13}$	$33.73_{\pm 3.61}$	$75.25_{\pm 0.63}$	$3.38_{\pm 0.15}$	$71.60_{\pm 0.87}$	$30.73_{\pm 1.85}$	$91.99_{\pm 0.96}$	$73.54_{\pm 2.30}$	$88.40_{\pm 0.97}$	$41.04_{\pm 6.59}$
GHRN	$86.26_{\pm 2.32}$	$42.46_{\pm 7.11}$	$75.30_{\pm 0.42}$	$3.40_{\pm 0.10}$	$71.71_{\pm 0.91}$	$31.17_{\pm 1.70}$	$91.54_{\pm 1.04}$	$71.55_{\pm 3.21}$	$88.37_{\pm 1.03}$	$40.77_{\pm 6.53}$
PC-GNN	$86.27_{\pm 2.13}$	$25.28_{\pm 4.26}$	$70.75_{\pm 0.09}$	$2.48_{\pm 0.19}$	$77.99_{\pm 0.84}$	$39.91_{\pm 1.76}$	$91.66_{\pm 0.79}$	$71.27_{\pm 4.77}$	$88.08_{\pm 2.44}$	$45.16_{\pm 6.97}$
ConsisGAD	$94.01_{\pm 0.22}$	$46.68_{\pm 0.61}$	$73.80_{\pm 0.22}$	$3.26_{\pm 0.08}$	$82.85_{\pm 0.48}$	$45.96_{\pm 1.15}$	$\mathbf{95.33}_{\pm 0.63}$	$\mathbf{86.33}_{\pm 0.65}$	$\mathbf{91.89}_{\pm 1.07}$	$\mathbf{80.39}_{\pm 1.67}$
CRoC	$\mathbf{95.58}_{\pm 0.52}$	$\mathbf{64.24}_{\pm 6.06}$	$\mathbf{76.62}_{\pm 0.54}$	$\mathbf{3.75}_{\pm 0.13}$	$\mathbf{83.57}_{\pm 0.87}$	$\mathbf{50.15}_{\pm 1.98}$	$92.44_{\pm 0.67}$	$76.48_{\pm 2.03}$	$89.20_{\pm 1.30}$	$52.98_{\pm 7.12}$

Table 4. The AUC of CRoC(GIN) with different variants.

RJA	L_{ctr}	L_{ce}^{crf}	Yelp (1%)	T-Soc (0.01%)	DGraph (1%)
✗	✗	✗	$73.28_{\pm 1.26}$	$87.86_{\pm 4.97}$	$71.13_{\pm 0.89}$
✗	✗	✓	$72.96_{\pm 1.39}$	$93.03_{\pm 0.45}$	$71.73_{\pm 0.71}$
✗	✓	✗	$73.89_{\pm 1.61}$	$90.85_{\pm 1.99}$	$71.75_{\pm 0.90}$
✗	✓	✓	$75.00_{\pm 1.45}$	$93.11_{\pm 0.66}$	$72.15_{\pm 0.73}$
✓	✗	✗	$79.40_{\pm 1.69}$	$82.18_{\pm 3.18}$	$74.36_{\pm 0.58}$
✓	✗	✓	$81.90_{\pm 0.83}$	$94.34_{\pm 1.15}$	$75.07_{\pm 0.91}$
✓	✓	✗	$82.97_{\pm 1.10}$	$90.54_{\pm 2.79}$	$75.16_{\pm 0.73}$
✓	✓	✓	$83.57_{\pm 0.87}$	$95.58_{\pm 0.52}$	$75.42_{\pm 0.75}$

- With limited supervision, other methods may fall into overfitting. In contrast, the capability to learn from unlabeled data endows our model with better generalization ability. This advantage is particularly pronounced on large-scale graphs, where CRoC can utilize more unlabeled data for training.
- Compared with ConsisGAD, which also focuses on limited supervision scenarios, CRoC shows some distinct advantages: (1) To incorporate unlabeled nodes for training, ConsisGAD introduces a learnable module for data augmentation. In contrast, CRoC's context refactoring mechanism is a parameter-free data augmentation method that adapts to GAD scenarios, resulting in lower computational complexity. (2) ConsisGAD uses predicted pseudo-labels as supervision signals for consistency training, which inevitably introduces noise even though they go through a filtering process. Different from ConsisGAD, CRoC always uses the given limited labels as the only supervised signal, avoiding being misled by potential camouflage. This makes CRoC perform significantly better than ConsisGAD in hard cases like DGraph.

We also observe that the performance gap of all models is minor in Amazon and T-Fin. In some cases, feature-based methods even outperform GNN models, aligning with findings from GADBench [25]. This suggests that graph structural information may have limited utility for anomaly detection in these datasets. In other words, node features may play a much more important role in decision-making. Consequently, these datasets may not be ideal benchmarks for evaluating specialized GAD models. We may focus on other datasets for more reliable GAD evaluations.

5.3 Ablation Studies

We further investigate the effectiveness of each proposed module. We conduct experiments by enabling/ disabling the RJA, L_{ctr} and L_{ce}^{crf}, which correspond to the Relation-aware Joint Aggregation, node-wise contrastive learning, and context refactoring. All other experimental settings are kept the same as above.

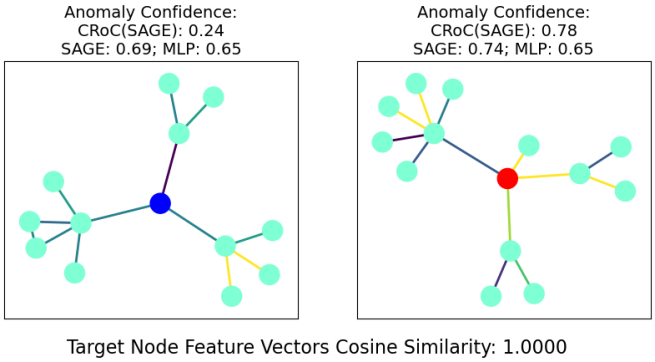

Figure 3. A case study in DGraph: blue (normal) and red (anomalous) nodes are the target nodes. Cyan nodes are neighboring nodes around the target node. Edge color differs for different relations.

Results of CRoC using GIN as the backbone are listed in Table 4. More extended results are provided in Appendix D.3 [35]. The findings indicate that most modules independently enhance the backbone model's performance. One exception is RJA in T-Soc, where the performance declines significantly if it is solely equipped. This is because RJA introduces additional learnable parameters to encode relations, which demands more data for training. Without context refactoring and contrastive learning, the model can only learn from the set of labeled nodes (and their neighborhood), which are extremely limited in T-Soc and result in overfitting. When integrating RJA with L_{ctr} or L_{ce}^{crf}, the performance is improved significantly for all cases. In CRoC, context refactoring and the contrastive learning strategy actively incorporate more unlabeled nodes for training. With abundant data to learn from, we can apply a more sophisticated model for GAD, even if the labels available for supervision are limited. This unleashes the learning capacity of the model, enabling it to learn more comprehensive embeddings for downstream tasks. In addition, as shown in Table 4, combining L_{ctr} and L_{ce}^{crf} usually leads to a better performance than applying them individually. This fact reflects that the knowledge learned from these two modules can be complementary to each other, making the model more powerful in GAD.

5.4 Qualitative Analysis

We also present some visualization results for qualitative analysis. Fig. 3 shows a potential camouflage case in DGraph, where anomalous and normal nodes have the same set of node features (cosine similarity is 1). In such cases, feature-based algorithms, such as MLP or XGBoost, will definitely fail. Although common GNNs, such as

Table 5. The averaged time cost (s) of running an update step. GHRN is run on CPU in T-Soc due to GPU out-of-memory.

Dataset	Amazon	Yelp	T-Fin	T-Soc	DGraph
GIN	0.04	0.08	0.12	0.50	0.18
PCGNN	0.03	0.03	0.05	3.02	0.61
BWGNN	0.03	0.05	0.12	3.97	1.12
GHRN	0.25	0.23	0.86	18.00 (CPU)	1.42
ConsisGAD	7.29	5.88	11.47	4.21	3.52
CRoC(GIN)	0.18	0.33	0.21	1.27	0.39

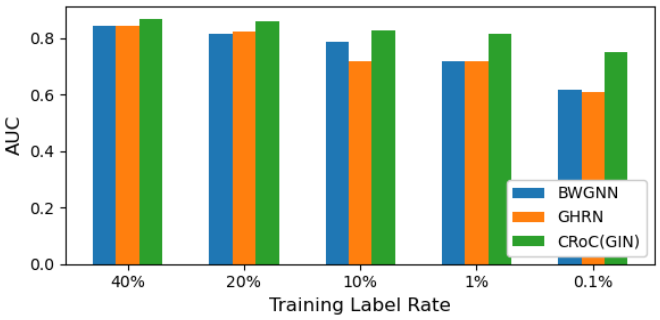

Figure 4. Experimental results for different training label rate settings.

SAGE [10], can learn additional information from the neighborhood of the target, they are unaware of the interaction types between nodes. Consequently, these models struggle to differentiate between two cases with similar local graph structures, just as shown in Fig. 3. In contrast, by introducing RJA into the aggregator, the GNN backbone can encode different relations of interactions around the target. This enables the model to generate a distinct embedding for each node, resulting in better performance in tricky cases, such as behavior camouflage. More visualization results can be found in Appendix D.4 [35].

5.5 *Analysis on Computational Efficiency*

CRoC is a plug-and-play scheme to enhance GNN models on GAD tasks, with little additional overhead over the original GNN backbone. Consider a (message-passing) GNN backbone with L layers, each outputting node embeddings with hidden dimension F. Denote N and M the number of nodes and edges in the graph, respectively. The overall time complexity of the backbone of a forward step is $O(L(\frac{M}{F}+N)F^2)$. After applying CRoC, the time complexity of the corresponding backbone is approximately $O(L(M+N)F^2)$. When the graph is sparsely connected (i.e., M and N in the same order, such as DGraph), the CRoC-enhanced GNN maintains the same order of time complexity as the original backbone. A detailed analysis is provided in Appendix B.3 [35].

We record the time cost of running an update (forward and backward) step of different methods on the same device, as shown in Table 5. CRoC(GIN) demonstrates linear scalability relative to its backbone (GIN). Moreover, CRoC(GIN) is significantly faster than other specialized GAD methods in large-scale graphs such as T-Soc and DGraph, highlighting its efficiency advantages.

5.6 *Results on Other Experimental Settings*

5.6.1 *Different Training Label Rate Settings*

We also investigate the performance of CRoC(GIN) when different numbers of labeled samples are available for training. Experimental results on Yelp are shown in Fig. 4. We can observe that CRoC(GIN) consistently outperforms other methods. When fewer training labels are available, the performance of CRoC(GIN) only drops slightly, while other methods suffer from a performance collapse. Such an observation implies that CRoC can be effective for different GAD scenarios (corresponding to cases where budgets for annotating training data are different). More experimental results and analyses can be found in Appendix D.5 [35].

Table 6. Experimental results (in AUC) on four GAD datasets.

Dataset	Yelp (1%)	Amazon (1%)	Weibo (1%)	Reddit (1%)
GAD-NR	54.36	70.00	87.71	57.99
TAM	56.43	70.64	N/A	**60.23**
CRoC(GIN)	**81.64**	**89.65**	**91.52**	59.92

Table 7. Experimental results (in Macro-F1) on four GAD datasets.

Dataset	Yelp (1%)	Amazon (1%)	T-Fin (1%)	T-Soc (0.01%)
BSL	62.83	**91.42**	**86.10**	75.67
CRoC(GIN)	**70.87**	74.81	84.26	**80.80**

5.6.2 *Different Evaluation Protocols*

There are many specialized GAD models other than the works listed in Table 3. However, due to variations in evaluation protocols (e.g., different datasets, train/test splits, assessment metrics, etc.), it is unable to directly include them [11, 21, 20, 41, 3, 19, 38] in Table 3. To ensure fair comparisons, we aligned CRoC's evaluation protocol with their settings and conducted additional experiments.

A part of the results is presented in Tables 6 and 7. We can observe that CRoC(GIN) performs better than other specialized GAD models in most cases. The advantage of CRoC is more significant when it is applied to large-scale graphs, such as Yelp and T-Soc, which demonstrates the superiority of the proposed framework. We refer to Appendix D.7 [35] for more detailed comparison results.

5.7 *Extended Experiments and Analyses*

Due to space limitations, other experiments and analyses, such as hyper-parameter investigations, are omitted from the main text. More explorations and experimental results refer to Appendix D [35].

6 Conclusions

In this paper, we propose Context Refactoring Contrast (CRoC) to enhance GNN models for Graph Anomaly Detection (GAD) tasks. We start with an analysis of some key challenges of GAD tasks and deal with the problems of camouflage and limited supervision. Different from previous works, we make use of the class imbalance of GAD and propose context refactoring to augment the original graph. The objective derived from context refactoring effectively correlates the limited labeled data with the abundant unlabeled data, enabling a GNN to learn more powerful representations by observing more data. In addition, we propose a relation-aware joint aggregation scheme to enhance the representation power of common GNNs on multi-relation graphs. By integrating context refactoring with contrastive learning, CRoC-enhanced models can further capture more diverse contextual patterns from the entire graph to compensate for the shortage of supervised signals. Extensive experiments on different scales of real-world GAD datasets demonstrate the superiority of CRoC over existing works under limited supervision conditions.

Acknowledgements

This work is supported in part by the Innovation and Technology Commission (ITS/244/16) and the CUHK MobiTeC R&D Fund.

References

[1] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[2] D. Chen, Y. Lin, W. Li, P. Li, J. Zhou, and X. Sun. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 3438–3445, 2020.

[3] J. Chen, G. Zhu, C. Yuan, and Y. Huang. Boosting graph anomaly detection with adaptive message passing. In *The Twelfth International Conference on Learning Representations*, 2024.

[4] N. Chen, Z. Liu, B. Hooi, B. He, R. Fathony, J. Hu, and J. Chen. Consistency training with learnable data augmentation for graph anomaly detection with limited supervision. In *The Twelfth International Conference on Learning Representations*, 2024.

[5] T. Chen, T. He, M. Benesty, V. Khotilovich, Y. Tang, H. Cho, K. Chen, R. Mitchell, I. Cano, T. Zhou, et al. Xgboost: extreme gradient boosting. *R package version 0.4-2*, 1(4):1–4, 2015.

[6] Y. Dou, Z. Liu, L. Sun, Y. Deng, H. Peng, and P. S. Yu. Enhancing graph neural network-based fraud detectors against camouflaged fraudsters. In *Proceedings of the 29th ACM international conference on information & knowledge management*, pages 315–324, 2020.

[7] W. Feng, J. Zhang, Y. Dong, Y. Han, H. Luan, Q. Xu, Q. Yang, E. Kharlamov, and J. Tang. Graph random neural networks for semi-supervised learning on graphs. *Advances in neural information processing systems*, 33:22092–22103, 2020.

[8] Y. Gao, X. Wang, X. He, Z. Liu, H. Feng, and Y. Zhang. Alleviating structural distribution shift in graph anomaly detection. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, pages 357–365, 2023.

[9] Y. Gao, X. Wang, X. He, Z. Liu, H. Feng, and Y. Zhang. Addressing heterophily in graph anomaly detection: A perspective of graph spectrum. In *Proceedings of the ACM Web Conference 2023*, pages 1528–1538, 2023.

[10] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[11] J. He, Q. Xu, Y. Jiang, Z. Wang, and Q. Huang. Ada-gad: Anomaly-denoised autoencoders for graph anomaly detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 8481–8489, 2024.

[12] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

[13] M. Jin, Y. Liu, Y. Zheng, L. Chi, Y.-F. Li, and S. Pan. Anemone: Graph anomaly detection with multi-scale contrastive learning. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 3122–3126, 2021.

[14] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[15] Q. Li, Y. He, C. Xu, F. Wu, J. Gao, and Z. Li. Dual-augment graph neural network for fraud detection. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 4188–4192, 2022.

[16] Y. Liu, X. Ao, Z. Qin, J. Chi, J. Feng, H. Yang, and Q. He. Pick and choose: a gnn-based imbalanced learning approach for fraud detection. In *Proceedings of the web conference 2021*, pages 3168–3177, 2021.

[17] Z. Liu, Y. Dou, P. S. Yu, Y. Deng, and H. Peng. Alleviating the inconsistency problem of applying graph neural network to fraud detection. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*, pages 1569–1572, 2020.

[18] X. Ma, J. Wu, S. Xue, J. Yang, C. Zhou, Q. Z. Sheng, H. Xiong, and L. Akoglu. A comprehensive survey on graph anomaly detection with deep learning. *IEEE Transactions on Knowledge and Data Engineering*, 35(12):12012–12038, 2021.

[19] X. Ma, R. Li, F. Liu, K. Ding, J. Yang, and J. Wu. Graph anomaly detection with few labels: A data-centric approach. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2153–2164, 2024.

[20] H. Qiao and G. Pang. Truncated affinity maximization: One-class homophily modeling for graph anomaly detection. *Advances in Neural Information Processing Systems*, 36:49490–49512, 2023.

[21] A. Roy, J. Shu, J. Li, C. Yang, O. Elshocht, J. Smeets, and P. Li. Gad-nr: Graph anomaly detection via neighborhood reconstruction. In *Proceedings of the 17th ACM international conference on web search and data mining*, pages 576–585, 2024.

[22] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling. Modeling relational data with graph convolutional networks. In *The semantic web: 15th international conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, proceedings 15*, pages 593–607. Springer, 2018.

[23] K. Sun, Z. Lin, and Z. Zhu. Multi-stage self-supervised learning for graph convolutional networks on graphs with few labeled nodes. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 5892–5899, 2020.

[24] J. Tang, J. Li, Z. Gao, and J. Li. Rethinking graph neural networks for anomaly detection. In *International Conference on Machine Learning*, pages 21076–21089. PMLR, 2022.

[25] J. Tang, F. Hua, Z. Gao, P. Zhao, and J. Li. Gadbench: Revisiting and benchmarking supervised graph anomaly detection. In *Advances in Neural Information Processing Systems*, volume 36, pages 29628–29653, 2023.

[26] S. Thakoor, C. Tallec, M. G. Azar, M. Azabou, E. L. Dyer, R. Munos, P. Veličković, and M. Valko. Large-scale representation learning on graphs via bootstrapping. *arXiv preprint arXiv:2102.06514*, 2021.

[27] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[28] P. Velickovic, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm. Deep graph infomax. *ICLR (Poster)*, 2(3):4, 2019.

[29] Y. Wang, J. Zhang, S. Guo, H. Yin, C. Li, and H. Chen. Decoupling representation learning and classification for gnn-based anomaly detection. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*, pages 1239–1248, 2021.

[30] B. Wu, X. Yao, B. Zhang, K.-M. Chao, and Y. Li. Splitgnn: Spectral graph neural network for fraud detection against heterophily. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 2737–2746, 2023.

[31] S. Xiang, M. Zhu, D. Cheng, E. Li, R. Zhao, Y. Ouyang, L. Chen, and Y. Zheng. Semi-supervised credit card fraud detection via attribute-driven graph representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 14557–14565, 2023.

[32] S. Xiang, M. Zhu, D. Cheng, E. Li, R. Zhao, Y. Ouyang, L. Chen, and Y. Zheng. Semi-supervised credit card fraud detection via attribute-driven graph representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 14557–14565, 2023.

[33] S. Xie, W. C. Lau, et al. Cocos: Enhancing semi-supervised learning on graphs with unlabeled data via contrastive context sharing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 4272–4280, 2022.

[34] S. Xie, D. S. H. Tam, and W. C. Lau. Violin: virtual overbridge linking for enhancing semi-supervised learning on graphs with limited labels. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, pages 4451–4459, 2023.

[35] S. Xie, D. S. H. Tam, and W. C. Lau. Appendix for croc: Context refactoring contrast for graph anomaly detection with limited supervision. *arXiv preprint*, 2025. URL https://arxiv.org/abs/2508.12278.

[36] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[37] Z. Xu, L. Li, H. Li, Q. Sun, S. Hu, and R. Ji. Self-supervised graph representation learning for black market account detection. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, pages 330–338, 2023.

[38] H. Yu, Z. Liu, and X. Luo. Barely supervised learning for graph-based fraud detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 16548–16557, 2024.

[39] G. Zhang, J. Wu, J. Yang, A. Beheshti, S. Xue, C. Zhou, and Q. Z. Sheng. Fraudre: Fraud detection dual-resistant to graph inconsistency and imbalance. In *2021 IEEE international conference on data mining (ICDM)*, pages 867–876. IEEE, 2021.

[40] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[41] Q. Zhou, Y. Chen, Z. Xu, Y. Wu, M. Pan, M. Das, H. Yang, and H. Tong. Graph anomaly detection with adaptive node mixup. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 3494–3504, 2024.